APPLICATION NOTE

AN_262

FT_APP_FT_CLOCKS

Version 1.1

Document Reference No.: FT_000907

Issue Date: 2013-11-01

This document is to introduce the FT Clock Demo Application. The objective of the Demo Application is to enable users to become familiar with the usage of the FT800, the design flow, and display list used to design the desired graphical user interface or visual effect.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This design example demonstrates an interactive clock using lines, points, and Jpeg decode plus audio playback on an FT800 platform.

In the FT_clock application, a swiss-clock graphic with major and minor segments and three clock hands is the desired graphic. Further alternatives for the back-ground for the clock face will be shown (plain gradient face or a face generated from a jpeg image in the case where memory storage is readily available).The clock time can be adjusted by touching the minutes hand and using rotary movements. FT clock is an application that demonstrates the use of FT800 graphical primitives to construct a more complex image. This application utilizes all the features of the FT800 i.e. touch, audio and display. Note that this clock is distinct in nature and not the clock widget that is available in the FT800 widget library. In the FT Clock, three processes, audio playback, image decoding and display list construction are handled. Loading of image and audio, along with the construction of the display list will be as follow:

- Image uploading : Via the CMD FIFO by Load image command
- Display list : directly in DL RAM
- Audio play : via SPI to GRAM

1.1 Overview

The document will provide information on the image creation, tagging of audio and touch capabilities, and the structure of displays lists. Further, the application note will outline the general steps of the design flow, including display list creation, and integrating the display list with the system host micro-controller.

This document should be read in conjunction with the source code which can be found in section 4 or at:

http://www.ftdichip.com/Support/SoftwareExamples/FT800_Projects.htm

1.2 Scope

This document can be used as a guide by designers to develop GUI applications by using FT800 with any MCU via SPI or I^2C. Note detailed documentation is available on www.ftdichip.com/EVE.htm including:

- FT800 datasheet
- Programming Guide covering EVE command language
- AN_240 FT800 From the Ground Up
- AN_245 VM800CB_SampleApp_PC_Introduction - covering detailed design flow with a PC and USB to SPI bridge cable
- AN_246 VM800CB_SampleApp_Arduino_Introduction – covering detailed design flow in an Arduino platform
- AN_252 FT800 Audio Primer

2 Display Requirements

This section describes some of the key components of the design.

2.1 Wall Background image

In the FT Clock application, the wall background is a JPEG image. There are 10 JPEG images (bgimg0.jpg – bgimg9.jpg) which may be loaded as the application cycles round. The actual clock is overlaid to the coordinates and size matching the clock in the JPEG.

To change the background displayed touch the display in the background area. Two bitmap handles are used to assist swapping the images.

2.2 Clock face

The clock face (skin) is generated with FT800 primitives such as LINES and FTPOINTS as opposed to the clock widget, allowing for customised hands such as those on a Swiss clock to be generated. The face can be changed by tapping on the clock. This will change the colour of the face background and the hands.

2.3 Date and time

Time in the application is initialised to System time from the host controller. The date, month and year can be adjusted by touching the date display. Adjustment in date, month and year can be done only for the first two skins.

Time adjustment can be done using the tracker for the hour needle and the minute needle is adjusted accordingly. As the tracker is synced with System time, the second hand movement changes accordingly.

2.4 Audio for changing hour

In this application, the time change per hour is indicated by the recorded voice message. When the time changes past the hour, the recorded voice message reports the hour displayed. The audio files are uLAW encoded from a PC application such as aud_cvt from the FTDI utilities web page. There are 12 files (1.ULW – 12.ULW) – one for each hour.

2.5 Without SD card

If the SD Card is not present, the system can show only two wall backgrounds, and the audio alert is disabled.

NOTE: If using an FTDI MPSSE cable then the SD card is not required.

3 Design Flow

Every EVE design follows the same basic principles as highlighted in Figure 3.1.

The active list and the edited list which are continually swapped to update the display.Select and configure your host port for controlling the FT800 then wake the device before configuring the display. The creative part then revolves around the generation of the display list. There will be two lists. The active list and the updated/next list are continually swapped to render the display. Note, header files map the pseudo code of the design file of the display list to the FT800 instruction set, which is sent as the data of the SPI (or I²C) packet (typically <1KB). As a result, with EVE's object oriented approach, the FT800 is operating as an SPI peripheral while providing full display, audio, and touch capabilities.

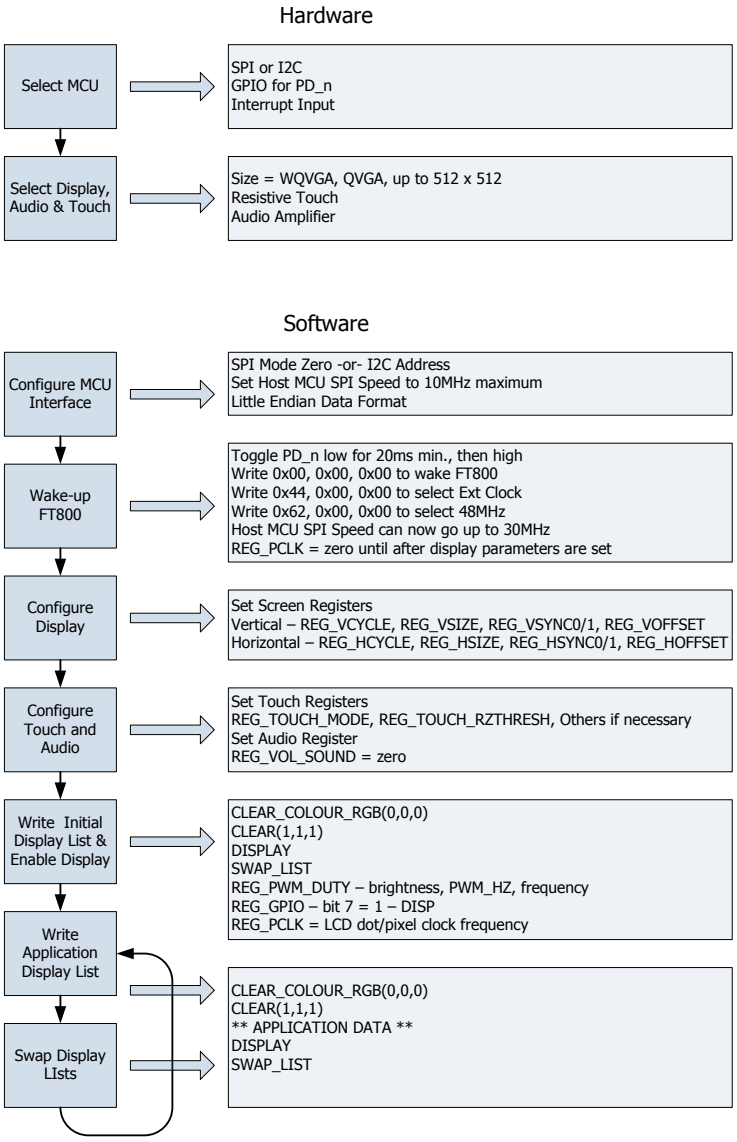

Figure 3.1 Generic EVE Design Flow

3.1 FtClocks Flowchart

The flow chart below is specific to the FtClocks application. The application uses internal Graphic RAM and SD card storage for storing bitmaps and audio files that may be seen and heard as the application plays.

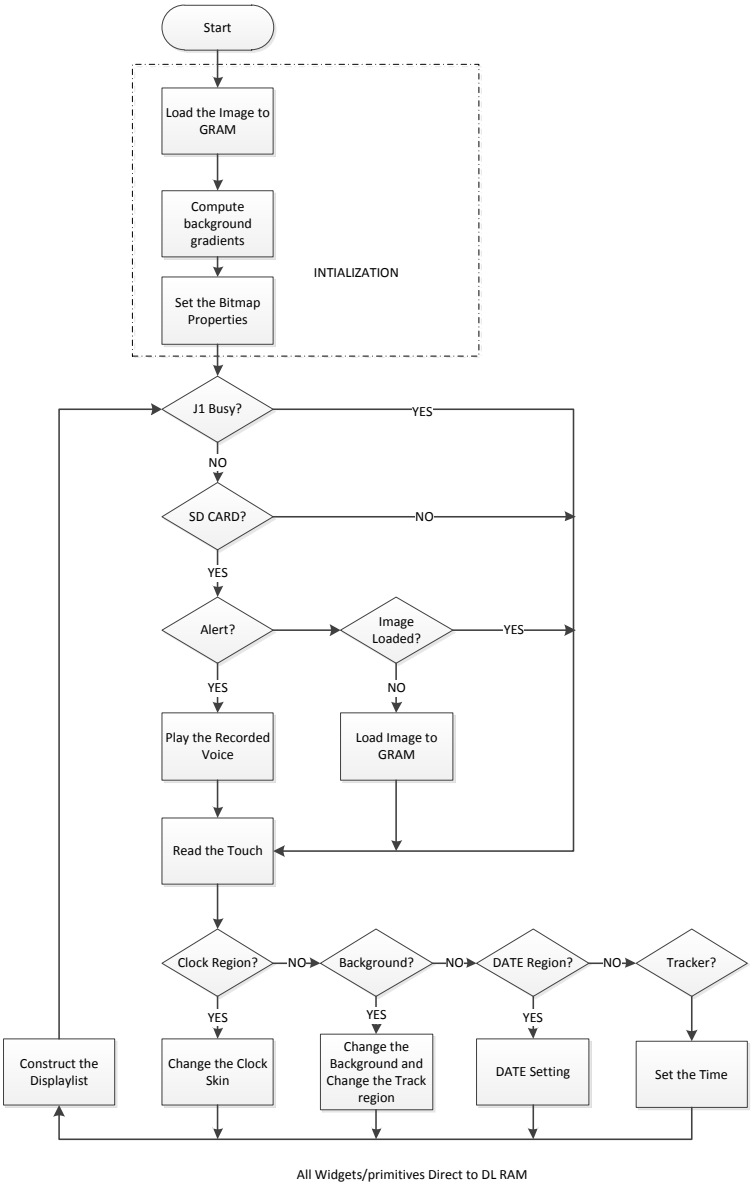

Figure 3.2 Flowchart

4 Description of the Functional Blocks

4.1 System Initialisation

Configuration of the SPI master port is unique to each controller – different registers etc, but all will require data to be sent Most Significant Bit (MSB) first with a little endian format.

The function labelled Ft_BootupConfig in this project is generic to all applications and will start by toggling the FT800 PD# pin to perform a power cycle.

```
/* Do a power cycle for safer side */
Ft_Gpu_Hal_Powercycle(phost,FT_TRUE);
Ft_Gpu_Hal_Rd16(phost,RAM_G);

/* Set the clk to external clock */
Ft_Gpu_HostCommand(phost,FT_GPU_EXTERNAL_OSC);
Ft_Gpu_Hal_Sleep(10);


/* Switch PLL output to 48MHz */
Ft_Gpu_HostCommand(phost,FT_GPU_PLL_48M);
Ft_Gpu_Hal_Sleep(10);

/* Do a core reset for safer side */
Ft_Gpu_HostCommand(phost,FT_GPU_CORE_RESET);

/* Access address 0 to wake up the FT800 */
Ft_Gpu_HostCommand(phost,FT_GPU_ACTIVE_M);
```

The internal PLL is then given a prompt by setting the clock register and PLL to 48 MHz.

Note 36MHz is possible but will have a knock on effect for the display timing parameters.

A software reset of the core is performed followed by a dummy read to address 0 to complete the wake up sequence.

The FT800 GPIO lines are also controlled by writing to registers:

```
Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0x80 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO_DIR));
Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x080 | Ft_Gpu_Hal_Rd8(phost,REG_GPIO));
```

And these allow the display to be enabled.

To confirm the FT800 is awake and ready to start accepting display list information the identity register is read in a loop until it reports back 0x7C. I will always be 0x7C if everything is awake and functioning correctly.

```
ft_uint8_t chipid;
//Read Register ID to check if FT800 is ready.
chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
while(chipid != 0x7C)
       chipid = Ft_Gpu_Hal_Rd8(phost, REG_ID);
```

Once the FT800 is awake the display may be configured through 13 register writes according to its resolution. Resolution and timing data should be available in the display datasheet.

```
    Ft_Gpu_Hal_Wr16(phost, REG_HCYCLE, FT_DispHCycle);
    Ft_Gpu_Hal_Wr16(phost, REG_HOFFSET, FT_DispHOffset);
    Ft_Gpu_Hal_Wr16(phost, REG_HSYNC0, FT_DispHSync0);
    Ft_Gpu_Hal_Wr16(phost, REG_HSYNC1, FT_DispHSync1);
    Ft_Gpu_Hal_Wr16(phost, REG_VCYCLE, FT_DispVCycle);
    Ft_Gpu_Hal_Wr16(phost, REG_VOFFSET, FT_DispVOffset);
    Ft_Gpu_Hal_Wr16(phost, REG_VSYNC0, FT_DispVSync0);
    Ft_Gpu_Hal_Wr16(phost, REG_VSYNC1, FT_DispVSync1);
    Ft_Gpu_Hal_Wr8(phost, REG_SWIZZLE, FT_DispSwizzle);
    Ft_Gpu_Hal_Wr8(phost, REG_PCLK_POL, FT_DispPCLKPol);
    Ft_Gpu_Hal_Wr8(phost, REG_PCLK,FT_DispPCLK);//after this display is visible on the
LCD
    Ft_Gpu_Hal_Wr16(phost, REG_HSIZE, FT_DispWidth);
    Ft_Gpu_Hal_Wr16(phost, REG_VSIZE, FT_DispHeight);
```

To complete the configuration the touch controller should also be calibrated

```
/* Touch configuration - configure the resistance value to 1200 - this value is
specific to customer requirement and derived by experiment */
    Ft_Gpu_Hal_Wr16(phost, REG_TOUCH_RZTHRESH,1200);
    Ft_Gpu_Hal_Wr8(phost, REG_GPIO_DIR,0xff);
        Ft_Gpu_Hal_Wr8(phost, REG_GPIO,0x0ff);
```

An optional step is present in this code to clear the screen so that no artefacts from bootup are displayed.

```
/*It is optional to clear the screen here*/
    Ft_Gpu_Hal_WrMem(phost, RAM_DL,(ft_uint8_t
                                        *)FT_DLCODE_BOOTUP,sizeof(FT_DLCODE_BOOTUP));
    Ft_Gpu_Hal_Wr8(phost, REG_DLSWAP,DLSWAP_FRAME);
```

4.2 Info()

This is a largely informational section of code and it starts by synchronising the physical xy coordinates of the displays touch layer with the displays visual layer.

A display list is started and cleared:

```
  Ft_Gpu_CoCmd_Dlstart(phost);
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
  Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
```

A text instruction is printed on the display followed by the call to the internal calibrate function:

```
   Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,26,OPT_CENTERX|OPT_CENTERY,"
   Please tap on a dot");
  Ft_Gpu_CoCmd_Calibrate(phost,0);
```

The display list is then terminated and swapped to allow the changes to take effect.

```
  Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
  Ft_Gpu_CoCmd_Swap(phost);
  Ft_App_Flush_Co_Buffer(phost);
  Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

Next up in the Info() function is the FTDI logo playback:

```
  Ft_Gpu_CoCmd_Logo(phost);
  Ft_App_Flush_Co_Buffer(phost);
  Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
  while(0!=Ft_Gpu_Hal_Rd16(phost,REG_CMD_READ));
  dloffset = Ft_Gpu_Hal_Rd16(phost,REG_CMD_DL);
  dloffset -=4;
  Ft_Gpu_Hal_WrCmd32(phost,CMD_MEMCPY);
  Ft_Gpu_Hal_WrCmd32(phost,100000L);
  Ft_Gpu_Hal_WrCmd32(phost,RAM_DL);
  Ft_Gpu_Hal_WrCmd32(phost,dloffset);
  play_setup();
```

A composite image with the logo and a start arrow is then displayed to allow the user to start the main application

```
do
  {
    Ft_Gpu_CoCmd_Dlstart(phost);
    Ft_Gpu_CoCmd_Append(phost,100000L,dloffset);
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_A(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_B(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_C(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_D(0));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_E(256));
    Ft_App_WrCoCmd_Buffer(phost,BITMAP_TRANSFORM_F(0));
    Ft_App_WrCoCmd_Buffer(phost,SAVE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(219,180,150));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(220));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(EDGE_STRIP_A));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(0,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F(FT_DispWidth*16,FT_DispHeight*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255));
    Ft_App_WrCoCmd_Buffer(phost,RESTORE_CONTEXT());
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(0,0,0));
   // INFORMATION
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,20,28,OPT_CENTERX|OPT_CENTERY,info[0]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,60,26,OPT_CENTERX|OPT_CENTERY,info[1]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,90,26,OPT_CENTERX|OPT_CENTERY,info[2]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,120,26,OPT_CENTERX|OPT_CENTERY,info[3]);
    Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight-
30,26,OPT_CENTERX|OPT_CENTERY,"Click to play");
    if(sk!='P')
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
    else
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(100,100,100));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(FTPOINTS));
    Ft_App_WrCoCmd_Buffer(phost,POINT_SIZE(20*16));
    Ft_App_WrCoCmd_Buffer(phost,TAG('P'));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2F((FT_DispWidth/2)*16,(FT_DispHeight-60)*16));
    Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(180,35,35));
    Ft_App_WrCoCmd_Buffer(phost,BEGIN(BITMAPS));
    Ft_App_WrCoCmd_Buffer(phost,VERTEX2II((FT_DispWidth/2)-14,(FT_DispHeight-
75),14,0));
    Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
    Ft_Gpu_CoCmd_Swap(phost);
    Ft_App_Flush_Co_Buffer(phost);
    Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
  }while(Read_Keys()!='P');
```

4.3 Memory Loading

The onboard graphics RAM is used to store the JPEG data and the audio data.

The file is copied from the PC (MPSSE version) or the SD card if present.

e.g. for the audio files, the function Load_afile is called with an address in the 256k Graphics RAM to be written to.

```
void Load_afile(ft_uint32_t add,  FILE *afile,ft_uint8_t noofsectors)
{
  ft_uint8_t tbuff[512],z;
  ft_uint16_t temp_add = 0;
  for(z=0;z<noofsectors;z++)
  {
    fread(tbuff,1,512,afile);
    temp_add = (add+(512L*z))&AUDIO_RAM_SPACE;
    Ft_Gpu_Hal_WrMem(phost,temp_add,tbuff,512L);
  }
```

4.4 Initial Clock Screen Setup

Compute the gradient background and load the image (SD card is present) to the GRAM via J1 and set the bitmap properties Image and the gradient.

This section of code defines the clock face and hands colour in a structure called clk_prp.

```
 // Intial settings
  r = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+0]);
  g = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+1]);
  b = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+2]);
  bgrgb =  ((ft_uint32_t)r<<16) | ((ft_uint32_t)g<<8) | ((ft_uint32_t)b<<0);
  r = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+4]);
  g = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+5]);
  b = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+6]) ;
  minrgb =  ((ft_uint32_t)r<<16) | ((ft_uint32_t)g<<8) | ((ft_uint32_t)b<<0);

  r = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+8]);
  g = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+9]);
  b = ft_pgm_read_byte(&clk_prp[(clkbgindex*12)+10]) ;
  secrgb =  ((ft_uint32_t)r<<16) | ((ft_uint32_t)g<<8) | ((ft_uint32_t)b<<0);
```

Where to draw the image and the time to display is then coded.

```
  temp_clk_x = image_prp[wall_index].clk_orgin_x;
  temp_clk_y = image_prp[wall_index].clk_orgin_y;
  temp_img_w = image_prp[0].imw;
  temp_img_h = image_prp[0].imh;
  temp_clk_s = image_prp[wall_index].clk_size;
  temp_img_x = (FT_DispWidth - temp_img_w)/2;
  temp_img_y = 5; // for 0//(FT_DispHeight - temp_img_h)/2;

  wall_bg = 0;
  bmp_handle=0;
  wall_index =0;

  Ft_Gpu_Hal_WrCmd32(phost,CMD_MEMZERO);
  Ft_Gpu_Hal_WrCmd32(phost,0);
  Ft_Gpu_Hal_WrCmd32(phost,256L*1024L);
```

This is followed by setting a gradient and drawing the background of the display

```
// for gradient effect
  for(tval=0;tval<(FT_DispHeight/2);tval++)
  {
    buff[FT_DispHeight-1-tval] = buff[tval] = (tval*0.9);
  }
  Ft_Gpu_Hal_WrMem(phost,4096L,buff,FT_DispHeight);

  for(tval=0;tval<40;tval++)
  {
    buff[tval] = tval*0.65;
  }
  Ft_Gpu_Hal_WrMem(phost,4368L,buff,40);
  for(tval=0;tval<177;tval++)
  {
    buff[tval] = tval/3;
  }
  Ft_Gpu_Hal_WrMem(phost,4408L,buff,177);

  incdec_setup();
  Init_Audio();
  Ft_DlBuffer_Index = 0;
  Ft_App_WrDlCmd_Buffer(phost,CLEAR(1,1,1));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_A(255));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(255,255,255));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_HANDLE(2));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SOURCE(4096L));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_LAYOUT(L8,1,FT_DispHeight));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, BORDER, FT_DispWidth,
FT_DispHeight));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_HANDLE(5));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SOURCE(4408L));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_LAYOUT(L8,1,177));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, BORDER, 284L, 177));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_HANDLE(3));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SOURCE(4368L));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_LAYOUT(L8,1,40));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SIZE(NEAREST, REPEAT, BORDER, 150, 40));
  Load_wallbg(bmp_handle);
  Ft_App_WrDlCmd_Buffer(phost,BEGIN(BITMAPS));
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2II(0,0,2,0));
  Ft_App_WrDlCmd_Buffer(phost,DISPLAY());
  Ft_App_Flush_DL_Buffer(phost);
  Ft_Gpu_DLSwap(DLSWAP_FRAME);
  Ft_CmdBuffer_Index = 0;

Ft_Gpu_CoCmd_Track(phost,temp_clk_x+temp_img_x,temp_clk_y+temp_img_y,1,1,CLOCK_MINTAG)
;
  Ft_App_Flush_Co_Buffer(phost);
                                  //L8 format
  Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
```

5 Operation

When the user compiles and runs the application code the first screen will be the calibration screen where the user must tap the screen in 3 places to align the touch and the display layers.

When the user compiles and runs the application code the first screen will be the calibration screen where the user must tap the screen in 3 places to align the touch and the display layers.

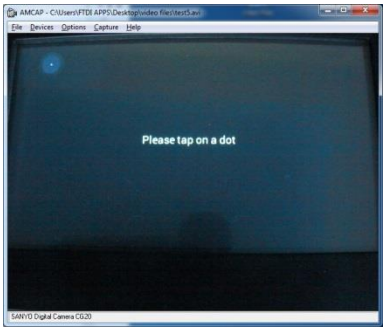

Figure 5.1 Tap Screen

This is followed by the logo and the composite logo/information screen which gives a short description of what the application does.

Figure 5.2 Composite screen

After pressing "Click to Play" the app displays the clock.

Figure 5.3 FtClock

5.1.1 Date Setting

If the user touches the date region on the screen, the date/month/year setting screen will appear, after setting the MCU waiting for the confirmation, the user should click again in date region date/month/year in the display.

Figure 5.4 Date Setting

5.1.2 Time Setting

The user can adjust the time by using the minute handle, when the minute handle is release the time gets stored in Global variable. The application can't change the system time.

5.1.3 Clock Skin

Clock skin colour is changeable, the user can change it by touching the clock region. The touch highlight will appear on the Clock.

5.1.4 Wall Background

The user can change the background image. The user can touch the background to change the background image. The touch highlight will appear on the image.

If there is no SD card, only two skins are available to change.

6 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited
(USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited
(Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited
(China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A– References

Document References

1. Datasheet for VM800C
2. Datasheet for VM800B
3. FT800 programmer guide FT_000793.
4. FT800 Embedded Video Engine Datasheet FT_000792

Acronyms and Abbreviations

Terms	Description
Arduino Pro	The open source platform variety based on ATMEL's ATMEGA chipset
EVE	Embedded Video Engine
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus

Appendix B – List of Tables & Figures

List of Figures

Appendix C– Revision History

Document Title: AN_262 FT_App_FT_Clocks

Document Reference No.: FT_000907

Clearance No.: FTDI# 357

Product Page: http://www.ftdichip.com/EVE.htm

Document Feedback: Send Feedback

Revision	Changes	Date
0.1	Initial draft release	2013-07-18
1.0	Version 1.0 updated wrt review comments	2013-08-21
1.1	Version 1.1	2013-11-01